An Experience Management System for a Software Consulting Organization

Carolyn Seaman[†*] cseaman@umbc.edu
Manoel Mendonça[§*] manoel@cs.umd.edu
Victor Basili[§*] basili@fc-md.umd.edu
Yong-Mi Kim[‡] yong-mi.kim@q-labs.com

[§]*University of Maryland at College Park*

[*]*Fraunhofer Center for Experimental Software Engineering*

[†]*University of Maryland at Baltimore County*

[‡]*Q-Labs, Inc.*

1 Introduction

Software is a major expense for most organizations and is on the critical path to almost all organizational activities. Individual software development organizations in general strive to develop higher quality systems at a lower cost for both their internal and external customers. Yet the processes used to develop such software are still very primitive in the way that experience is incorporated. Learning is often from scratch, and each new development team has to relearn the mistakes of its predecessors. Reuse of an organization's own products, processes, and experience is becoming more accepted as a feasible solution to this problem. But implementation of the idea, in most cases, has not gone beyond reuse of small-scale code components in very specific, well-defined, situations. True learning within a software development organization requires that organizational experiences, both technological and social, be analyzed and synthesized so that members of the organization can learn from them and apply them to new problems.

Suppose, for example, that a member of a software development group is considering the use of a particular software engineering technology on a forthcoming project. This member has heard that this technology has been used successfully in other projects in some other part of the organization, but cannot easily find out where or by whom. He or she would like very much to learn from the experiences of those previous projects, first to help make the decision to use the technology or not, then to help implement the technology in the current project. It would be helpful, obviously, to avoid the inevitable mistakes that are made the first time a new technology is tried. Also, it would be useful to see the costs of using that technology (e.g. the costs of new tools or training) in order to help estimate those costs for the current project. Without the organizational infrastructure to support access to previous experience from within the organization, this type of information would be very difficult, if not impossible, for the development team member to get.

This paper describes a system for supporting experience management in a multinational software improvement consultancy called Q-Labs. This Experience Management System (EMS) is based on the Experience Factory concept [1] proposed by Basili. This paper focuses on describing the design principles behind EMS and reports the results of an evaluation of its interface.

2 The Experience Factory

Basili proposed the Experience Factory as an organizational infrastructure to produce, store, and reuse experiences gained in a software development organization [1,2,3]. The Experience Factory idea organizes a software development enterprise into two distinct organizations, each specializing in its own primary goals. The Project Organization focuses on delivering the software product and the Experience Factory focuses on learning from experience and improving software development practice in the organization. Although the roles of the Project Organization and the Experience Factory are separate, they interact to support each other's objectives. As illustrated in Figure 1, the feedback between the two parts of the organization flows along well-defined channels for specific purposes. Also, the Experience Factory

supports the meta process defined by Basili's Quality Improvement Paradigm (QIP) [6]. As shown in Figure 1, for each new project: the problem at hand is characterized (1), goals are set (2), a suitable process is chosen (3), the process is executed and measured (4), outputs are analyzed (5), and lessons and products are packaged and stored in the experience base for future reuse (6).

Experience Factories recognize that improving software processes and products requires: (1) continual accumulation of evaluated and synthesized experiences in *experience packages*; (2) storage of the experience packages in an integrated *experience base* accessible by different parts of the organization; and (3) creation of *perspectives* by which different parts of the organization can look at the same experience base in different ways. Some examples of experience packages might be the results of a study investigating competing design techniques, a software library that provides some general functionality, or a set of data on the effort expended on several similar projects.

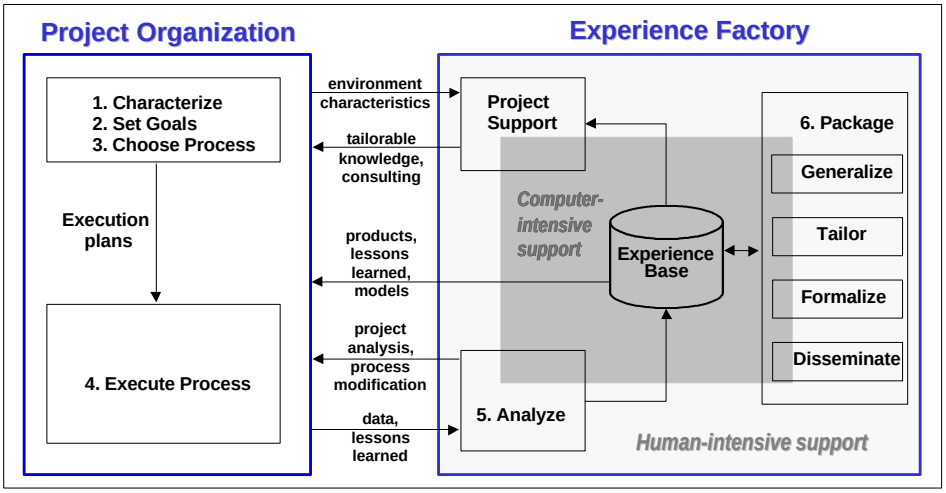

Figure 1. Experience Factory structure

The Experience Factory concept has been implemented in a number of software development organizations that have addressed the above questions in various ways (e.g. [4,5,9]). The Software Engineering Laboratory (SEL) [4] is an example of an Experience Factory. The SEL Quality Improvement Paradigm provides a practical method for facilitating product-based process improvement within a particular organization. Because it directly ties process improvement to the products produced, it allows an organization to optimize its process for the type of work that it does. Using this approach, the SEL has reduced development costs by 60%, decreased error rates by 85%, and reduced cycle time by 20% over the past 10 years. Establishing an Experience Factory, however, is a long-term endeavour requiring a great deal of commitment on the part of both management and development staff. Implementing an Experience Factory involves substantial up-front costs. It requires instilling a new philosophy of learning into an organization, establishing an organizational structure and processes for the Experience Factory to collect, package and share experiences. Once in place, it will also require substantial ongoing effort and commitment to maintain itself as an effective agent for continuous software process improvement.

We believe that emerging computing technologies – such as distributed systems, visual query interfaces, and intranets – offer great potential to support the establishment and maintenance of Experience Factories in organizations. This paper reports preliminary results and experiments from a research project aimed at implementing a system for supporting an Experience Factory within an industrial setting.

3 The Principles Behind the Experience Management System

We have found it useful to discuss the problem of software experience capture and reuse, and our approach to addressing it, in terms of the 3-layer conceptual view shown in Figure 2. This view shows three aspects of the problem, all of which need to be addressed before a complete solution can be implemented. At the lowest level, there are issues of how experience should be electronically stored in a repository and made accessible across geographical boundaries. The middle level deals with user interface issues, including how experiences are best presented to a user and how the user interacts with the automated system to

manipulate, search, and retrieve experience. At the top level, the organizational issues of how experience reuse will fit into the work of the organization, how the experience base will be updated and maintained, and how experiences will be analyzed and synthesized over time, are addressed. The bottom two levels of Figure 2 define the computer-intensive support pictured in Figure 1. The top level of Figure 2 defines the interface between the human-intensive and the computer-intensive areas described in Figure 1.

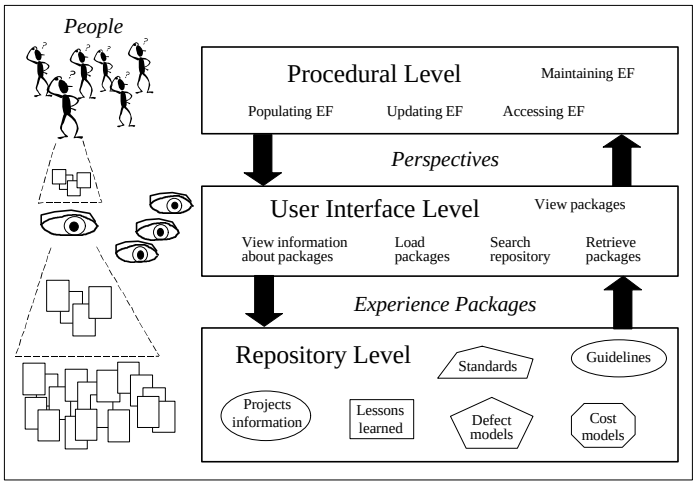

Figure 2. The three levels of an Experience Management System

Allied with this conceptual view, we have defined a set of requirements aimed at making the EMS reliable, easy to use, and flexible enough to support the Experience Factory concept.

R1. The system shall support geographically distributed organizations allowing them to share and manage experience packages remotely.

R2. The repository shall be robust, reliable, and portable to standard computer platforms.

R3. The user interface level shall be as platform independent as possible.

R4. The data model shall be simple but powerful enough to model diverse classes of "experience packages." The system will adapt to the current practices, processes, and products of different organizations, and not vice-versa.

R5. The system shall be easy to learn and self explanatory. The user interface shall be easy to use and the stored information shall be easy to search and retrieve.

This conceptual view, along with the requirements, form the basis of several ongoing efforts to implement experience management systems in a variety of settings. The first of these efforts, the Q-Labs EMS, is described in this paper. Lessons learned from our work with Q-Labs will be fed into other EMS efforts in the future.

4 The Q-Labs EMS

The Experimental Software Engineering Group (ESEG) at the University of Maryland and Q-Labs, Inc., have been working together for nearly three years on a project aimed at building the infrastructure to support a true Experience Factory within Q-Labs, resulting in an Experience Management System (from here on called the "Q-Labs EMS"). Q-Labs is a multi-national software engineering consulting firm that specializes in helping its clients improve their software engineering practices by implementing state-of-the-art technologies in their software development organizations. Q-Labs has helped many of its clients implement some of the principles of the Experience Factory. Q-Labs' objectives for this project have been to provide a "virtual office" for the organization, which is spread across two continents, and to allow each Q-Labs consultant to benefit from the experience of every other Q-Labs consultant.

4.1 System Architecture

In order to fulfill the first requirement presented in section 3, to support geographically distributed organizations, the Q-Labs EMS is a client-server system. The clients enforce the policies defined at the procedural level and implement the system front-end applications defined by the user interface level (here referring to the levels in Figure 2). The server implements the system repository. The architecture of the system is shown in Figure 3. It follows a three-tier model. At the top level, we have the EMS Manager and EMS Visual Query Interface (VQI) applications. They work as client applications sending requests to a "middle tier" of services. This "EMS Server" receives messages from the client applications and translates them into low-level SQL (Standard Query Language) calls to an "EMS Repository."

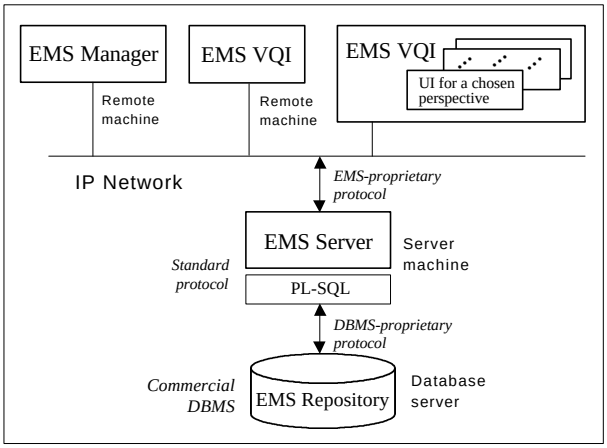

Figure 3. Q-Labs EMS architecture

In order to fulfill the second requirement, repository robustness and portability, the EMS Repository uses standard database technology. It stores all the information necessary for the EMS operation in a relational database managed by a commercial DBMS (Data Base Management System.) The link between the server and the repository is done through standard embedded SQL (PL-SQL.) This makes the repository portable to standard commercial DBMS, and virtually portable to any platform.

In order to fulfill the third requirement, a platform independent user interface, the client applications are implemented in Java™. This makes them portable to any platform that has a Java virtual machine. To date we have tested the client applications – EMS Manager and EMS VQI – on Unix, Windows (NT and 98), and Macintosh platforms.

4.2 Data Model

An early, crucial task in this project has been identifying the pieces of information that should be packaged as experience. However, each organization has different needs and experiences. In order to fulfill the fourth requirement, data model simplicity and flexibility, we introduce the concept of a **perspective**. A *perspective* defines a class of packages much like an *object class* in an object-oriented system. A perspective is defined by three parts: a classification part, a relationship part, and a body part.

The classification part, called the perspective **taxonomy**, defines a classification model for the packages instantiated from that perspective. The perspectives' taxonomies describe the contents of an experience base in an organization's own terminology, thus guiding users intuitively towards the experiences of interest to them. A taxonomy is composed of **attributes** with well-defined naming and typing. The attributes effectively define the facets that can to be filled by an experience packager to characterize a package instantiated in a given perspective.

The relationship part, called the perspective's **links**, defines the relationship between the packages instantiated in this perspective and other packages in the experience base. Like attributes, links have a name and type associated with them.

The perspective **body** defines the **elements** that compose the experience packages instantiated from this perspective. Like attributes and links, elements have names and types. The type is usually a file or a list of files. Those files are internally stored in the experience base as large objects when a package is instantiated from a perspective.

4.3 Visual Query Interface

In order to fulfill the fifth requirement, a search and retrieval interface that is easy to learn and self-explanatory, we adopted a visual query interface (VQI) concept. As proposed by Shneiderman [13], visual query interfaces let users "fly through" stored information by adjusting widgets and viewing animated results in the computer screen. In EMS, they allow easy interactive querying of the repository based on various attributes of the experience packages. Built in to the interface is the set of attributes defined for the perspective currently being viewed. Figure 5 shows the user interface for the Q-Labs EMS. Upon login a user will have a set of perspectives from which he/she can look at stored experience packages. A user will fire a VQI by selecting one of those perspectives. The VQI will display the packages that are associated with this perspective together with the attributes and query devices (slider bars, check boxes, etc.) used to search and browse those packages. The widgets used on the interface are defined on the fly based on the data types and number of different values associated with each attribute.

Using the VQI, the user interactively searches the experience packages associated with a certain perspective by manipulating the widgets on the right and observing the number of selected packages on the two-dimensional chart. Once a small subset of packages is selected using the VQI query devices, the user can quickly examine specific packages by clicking on them. This will fire a Web Page with a complete description of the selected package, including its links and elements. If the selected package corresponds to the user's expectations, he/she can click on the desired elements to retrieve the package's files.

The VQI has two features that we believe are fundamental to EMS. First, its search is interactive and controlled by the user. This allow the user to easily control the number of matches by widening or narrowing the search scope with a few mouse clicks. This is a clear advantage over keyword-based search – such as those executed by Worldwide Web search engines. We hypothesize that this significantly helps users to find packages that are useful to them even when an exact match is not available. The second key feature of this type of interface is that it allows people to visualize the amount of stored experience and the classification schema used by the organization. We believe that this significantly helps new users to get used to EMS and is also an important learning medium for new team members.

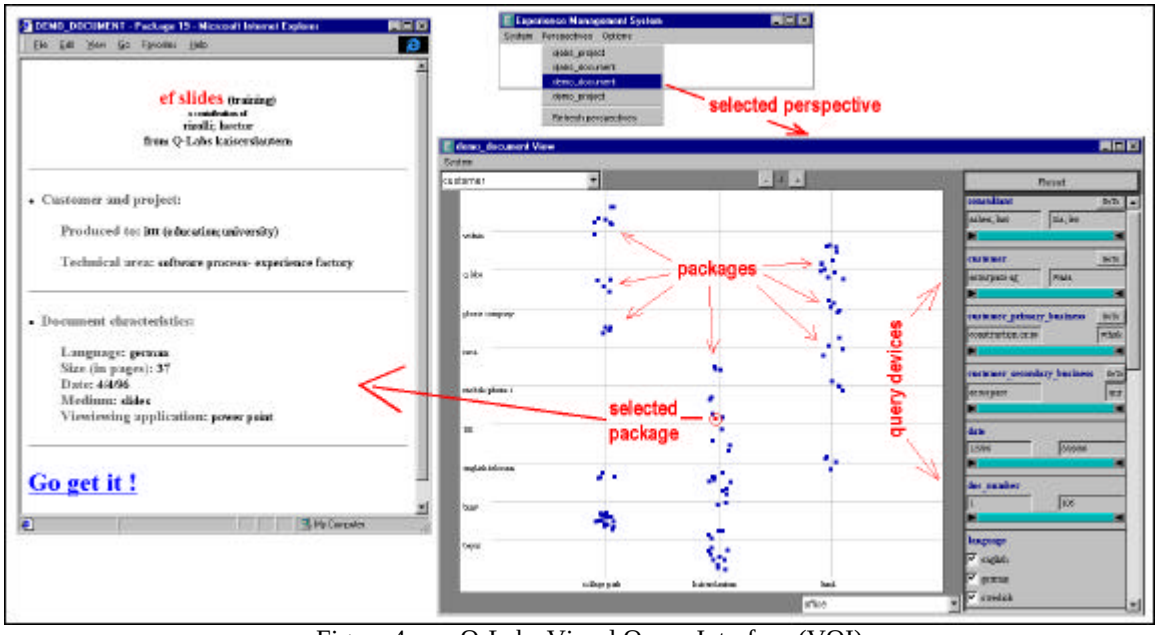

Figure 4. Q-Labs Visual Query Interface (VQI)

The user interface also has functionality to allow users to submit new experience packages to the experience base. This functionality uses the attributes, links, and elements associated with the perspectives to produce the forms that a user must complete to describe new packages.

5 Interface prototype evaluation

The first of several planned empirical studies to evaluate the Q-Labs EMS prototypes was an evaluation of the interface. This initial prototype consisted of the VQI (pictured in Figure 5), a simple data entry interface used to submit experience packages (just a form with each field corresponding to one of the defined attributes for a given perspective), and a small repository populated with a collection of real Q-Labs documents and project descriptions. Two perspectives were also provided with this prototype. The *documents* perspective used attributes of documents (e.g. author, date, title, etc.) as the search mechanisms, while the *projects* perspective used attributes of projects (e.g. project lead, customer, start date, finish date, total effort, etc.). Some attributes were common to both perspectives (e.g. technical area). The evaluation was carried out at this point in the project (before having a full working system) because it was essential to get user feedback on the basic paradigms we had chosen before we proceeded further.

5.1 Study design

The interface evaluation study was based on qualitative methods [10]. The importance of such methods in validating software engineering technology is discussed by Seaman in [12]. The goals of the interface evaluation study were:

1. To evaluate the current set of attributes (in both the "projects" and "documents" taxonomies) in terms of completeness, usefulness, and clarity.
2. To evaluate the visual query search and retrieval interface in terms of usefulness, usability, and appropriateness.
3. To evaluate the data entry interface in terms of feasibility, usability, and impact on working procedures.

These goals were refined into a set of questions that guided the design of the study. To answer these questions, two types of data were collected. The first data source consisted of detailed notes concerning how the subjects used the prototype and the comments they made while using it. The second data source came from a set of interviews that were conducted at the end of each evaluation session. The questions asked during the interviews are shown below in Figure 5.

1. What did you like most about the search and retrieval interface?
2. Was there anything really annoying about using it?
3. Was it easy to move around and do things with the mouse and keyboard?
4. Is there any information that would be useful to include in the interface that isn't there?
5. Are there any attributes that are not clear in their meaning?
6. What attributes did you use most in searches?
7. Did you feel that you were able to find what you were looking for using the interface?
8. How satisfied are you that tasks can be completed with the minimal number of steps?
9. How could the interface be improved?
10. Do you think you would use this tool, once the database was populated, in your everyday work? Does it support the way you normally work?
11. What did you like most about the data entry interface?
12. Was there anything really annoying about using it?
13. Were there any parts of the data entry interface where it wasn't clear what information you should enter?
14. How was using this interface different from the usual procedure for recording this type of information? Do you think, in general, that this would save you time or not?
15. How could the interface be improved?
16. Do the different parts of the system have consistent appearance and work in similar ways?
17. How satisfied are you with the system appearance in terms of color, layout, and graphics usage?

Figure 5. Evaluation Interview Questions

Interface evaluation sessions were held with five different Q-Labs consultants from three different offices in May and June of 1999. In each session, the subject was given a short hands-on training, then given a set of exercises that represented common Q-Labs work scenarios. The exercises were taken from the set of use cases we had collected as part of the initial requirements gathering activity for EMS. The subjects were asked to choose some of the exercises and then to use the Q-Labs EMS prototype to gain information

relevant to the scenario described in each exercise. They were also asked to verbalize their thoughts and motivations while working through the exercises. This technique, called a "think aloud" protocol [8], is often used in usability studies (and occasionally in other software engineering studies [14]) to capture a subject's immediate impressions, thought processes, and motivations while performing a task. The subjects could and did ask questions of the researcher conducting the session. After several exercises had been completed, a short interview was conducted, using the questions presented above as an interview guide. All the sessions were audiotaped and observed by at least one researcher. Each session lasted about 1.5 to 2 hours. Although the tapes were not transcribed verbatim, they were used to write very detailed notes after the fact.

The notes written from the tapes served as the major data source for the analysis part of the study. The analysis method used was the constant comparison method [7,10]. This method begins with coding the field notes by attaching codes, or labels, to pieces of text that are relevant to a particular theme or idea that is of interest in the study. Then passages of text are grouped into patterns according to the codes and subcodes they've been assigned. These groupings are examined for underlying themes and explanations of phenomena. The next step is the writing of a field memo that articulates a proposition (a preliminary hypothesis to be considered) or an observation synthesized from the coded data. In this case, the field memo written as part of this process became the results of the study, which are reported in the next section.

5.2 Results

The subjects generally liked the basic elements of the search and retrieval interface. In particular, they seemed to have no trouble mastering the search mechanism and liked how it was easy to negotiate the interface and see the distribution of packages among different attribute values. They also liked the immediate feedback in the graph part of the interface in response to changes made with the search mechanisms. Subjects were also able to glean useful information from the interface even when they couldn't find exactly what they were looking for. For example, one subject found a document that was not exactly what she wanted, but she saw the primary author's name and decided that would be a good contact, and so she felt she had found useful information.

The learning curve on the search and retrieval interface was fairly short. By the second or third exercise tried, all of the subjects were conducting their searches very rapidly and confidently. For some subjects, it was even quicker. Subjects generally narrowed their searches down to about 2-4 "hits" before looking at individual packages. This was seen as a "reasonable" number of packages to look through.

Several major annoyances surfaced during the evaluation. One was the use of slider bars. Several subjects had trouble figuring out the mechanics of using and interpreting them. Several subjects suggested using some form of checkboxes instead of the slider bars. Another annoyance had to do with the relationship between the two perspectives and the lack of linkage between them. After finding some relevant project in the projects perspective, subjects had to then start up the document perspective and start a search from scratch in order to find documents related to the project. A related problem was the confusion caused by some attributes and attribute values existing in one perspective but not the other.

As for the data entry interface, the data being collected was seen to be appropriate, but otherwise it left a lot to be desired. Subjects in general found the data entry interface unusable because they needed more guidance as to what attribute values to enter in the fields. Almost all of the subjects suggested pull-down menus or automatic fill-ins to decrease the amount of typing and increase consistency. In general, the subjects saw this interface as just a skeleton of what was needed.

All of this was valuable feedback that has been used in our plans for further development of the Q-Labs EMS. Although we knew that the interface we were evaluating was not ideal, we had not anticipated some of the specific problems that our subjects reported. For example, we had not considered the slider bar mechanism to be a problem, but our subjects definitely did. Also, although we knew the data entry interface needed some improvements (many of the suggestions from the subjects were already in our development plans), we had not considered it as completely unusable as our subjects did. On the other hand, the study validated some of our basic choices in the interface design, e.g. the VQI and the use of attributes and perspectives. Thus we can, with confidence, continue improvement of the interface without changing the underlying structure.

There were also some lessons learned about how the interface evaluation was conducted. Some problems came up related to the limited scope of the repository. Subjects were sometimes frustrated when there was nothing to be found for their search criteria. Subjects were also bothered by inconsistencies in the sample data. In particular, one subject found that there was a document in the documents perspective, that had a project name associated with it, but that project was not to be found in the projects perspective.

The interface evaluation, in general, proved to be a valuable and timely tool for getting feedback from the eventual users of the Q-Labs EMS. The effort involved was relatively small, although finding and scheduling subjects was difficult and caused some delays. Although much remains to be done before an operational system is delivered, the evaluation assured us that the Q-Labs EMS will eventually be acceptable to its intended users. In addition, the evaluation provided an opportunity to disseminate the aims of our project, and our work thus far, throughout Q-Labs.

6 Conclusions

We have described an ongoing project involving the Experimental Software Engineering Group (ESEG) at the University of Maryland and Q-Labs, Inc. that aims to provide a system (with both organizational and automated elements) to support software engineering experience capture and reuse. The current design of this system, called the Q-Labs EMS, is outlined, in particular its architecture and its user interface. Currently, an interface prototype exists and has been evaluated. This evaluation is described in detail. The results of the evaluation have assured us not only that the Q-Labs EMS will eventually be successfully deployed throughout Q-Labs, but will also serve as a testbed for our further investigation of software experience capture and reuse. However, much needs to be done before a working version of this system is in place. The prototype that has been evaluated encompassed only some of the automated features of the system. Much of the technical work remains, as well as the organizational part of the system. The latter includes designing, implementing, and evaluating new organizational procedures and deployment strategies to ensure the acceptance of EMS at Q-Labs.

7 References

[1] Basili, Victor R. Software Development: A Paradigm for the Future. *In Proc. of COMPSAC '89*, Orlando, Florida, pp. 471-485, September 1989.

[2] Basili, Victor R., and Gianluigi Caldiera, Improve Software Quality by Reusing Knowledge and Experience. *Sloan Management Review*, MIT Press, Volume 37, Number 1, Fall 1995.

[3] Basili, Victor R., Gianluigi Caldiera, and H. Dieter Rombach. The Experience Factory. In *Encyclopedia of Software Engineering*, New York: John Wiley & Sons, 1994. pp. 470-476.

[4] Basili, Victor R., Gianluigi Caldiera, Frank McGarry, Rose Pajerski, G. Page, and S. Waligora. The Software Engineering Laboratory - an Operational Software Experience Factory. *In Proceedings of the International Conference on Software Engineering*, May 1992, pp. 370-381.

[5] Basili, Victor, Michael K. Daskalantonakis, and Robert H. Yacobellis. Technology Transfer at Motorola. *IEEE Software*, March 1994, pp. 70-76.

[6] Basili, Victor, and H. Dieter Rombach. The TAME Project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758-773, June 1988.

[7] B.G. Glaser and A.L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company, 1967.

[8] J.T. Hackos and J.D. Redish. *User and Task Analysis for Interface Design*. New York:John Wiley and Sons, 1998, chapter 9, pp. 258-9.

[9] Houdek, F., K. Schneider, and E. Wieser (April 1998). Establishing Experience Factories at Daimler-Benz: An Experience Report. In *Proc. of 20th International Conference on Software Engineering*, Kyoto, Japan, pp. 443-447.

[10] M.B. Miles and A.M. Huberman. *Qualitative Data Analysis: An Expanded Sourcebook*, second edition, Thousand Oaks:Sage, 1994.

[11] Prieto-Diaz, R. Classifying of Reusable Modules. In T.J. Biggerstaff and A. Perlis, editors, *Software Reusability*, Volume I, ACM Press, 1990.

[12] Seaman, C.B. Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, 25(4):557-572, July/August 1999.

[13] Shneiderman, Ben. Dynamic Queries for Visual Information Seeking. *IEEE Software*, Vol. 6, No. 11, November 1994, pp 70-77.
[14] A. von Mayrhauser, and A.M. Vans. Identification of dynamic comprehension processes during large scale maintenance" *IEEE Transactions on Software Engineering*, 22(6):424-437, June 1996.